Benchmarking Numerical Multiobjective Optimizers Revisited

Dimo Brockhoff
Inria Lille – Nord Europe
DOLPHIN Team
Villeneuve d'Ascq, France
dimo.brockhoff@inria.fr

Thanh-Do Tran
Inria Lille – Nord Europe
DOLPHIN Team
Villeneuve d'Ascq, France
thanh-do.tran@inria.fr

Nikolaus Hansen
Inria Saclay – Ile-de-France
TAO Team
Orsay, France
nikolaus.hansen@lri.fr

ABSTRACT

Algorithm benchmarking plays a vital role in designing new optimization algorithms and in recommending efficient and robust algorithms for practical purposes. So far, two main approaches have been used to compare algorithms in the evolutionary multiobjective optimization (EMO) field: (i) displaying empirical attainment functions and (ii) reporting statistics on quality indicator values. Most of the time, EMO benchmarking studies compare algorithms for fixed and often arbitrary budgets of function evaluations although the algorithms are any-time optimizers. Instead, we propose to transfer and adapt standard benchmarking techniques from the single-objective optimization and classical derivative-free optimization community to the field of EMO. Reporting *target-based runlengths* allows to compare algorithms with varying numbers of function evaluations quantitatively. Displaying *data profiles* can aggregate performance information over different test functions, problem difficulties, and quality indicators. We apply this approach to compare three common algorithms on a new test function suite derived from the well-known single-objective BBOB functions. The focus thereby lies less on gaining insights into the algorithms but more on showcasing the concepts and on what can be gained over current benchmarking approaches.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*global optimization, unconstrained optimization*

Keywords

Benchmarking, Black-box optimization, Multiobjective optimization

1. A BRIEF OVERVIEW OF MULTIOBJECTIVE PERFORMANCE ASSESSMENT

Multiobjective optimization problems occur frequently in practice and several optimization algorithms are available that typically aim at finding a set of solutions that approximate the set of Pareto-optimal solutions with respect to the three criteria convergence, spread, and a good distribution [21]. In order to suggest well-performing algorithms to the practitioner, one needs to get insights into when and where algorithms have their strengths and weaknesses. Also for algorithm design, it is important to know where algorithms are failing in order to improve them while not introducing additional detrimental effects on the performance. Due to the difficulty of theoretical analyses, such investigations on algorithm performance are typically done by numerical experiments—in other words by *benchmarking*.

While in the early stages of (evolutionary) multiobjective optimization, visual comparisons of the resulting Pareto front approximations of algorithms have been used to draw conclusions, in recent years two complementary approaches have been developed to compare stochastic and set-based multiobjective optimization algorithms. The study of *Empirical Attainment Functions (EAFs)* [8], on the one hand, allows for graphical tools to visualize the (empirical) "probabilistic distribution of the outcomes obtained by a stochastic algorithm in the objective space" and the differences between algorithms in objective space [15]. On the other hand, most of the current studies in the EMO field use *quality indicators* such as the hypervolume- or ε-indicator to assess the quality of resulting solution sets [12, 20, 22]. Both the EAF approach and the idea of reporting quality indicators have in common that typical studies only report data for one or a few, often arbitrary, budgets of function evaluations[1].

Contrarily, in the single-objective blackbox optimization community, state-of-the-art benchmarking studies usually report (the distribution of) target-based runlengths[2] and illustrate them in so-called data and performance profiles [7, 16] which are also known as empirical cumulative distribution functions or empirical runtime distributions [11]. This has the advantage that the reported values are quantitative measurements on a ratio scale and, as runlengths, easily interpretable. For a specific problem difficulty, i.e., a target function value f_{target}, an algorithm $\mathcal{A}$ can be easily considered c times slower than an algorithm $\mathcal{B}$ if it takes c times longer to reach a function value of f_{target}. When comparing function (or indicator) values achieved at specific budgets,

[1] In the case of combinatorial problems, where the view of the optimization problem as a blackbox is often violated, we also see studies that fix the computation time.

[2] Given a target function value f_{target}, the target-based runlength of algorithm $\mathcal{A}$ on function f is the number of function evaluations until $\mathcal{A}$ samples a solution s with $f(s) < f_{\text{target}}$.

however, the interpretation of reported indicator values is (at most) not intuitive. Another advantage of reporting target-based runlengths is that it allows to compare algorithms with different stopping criteria more naturally.

Related Benchmarking Studies.

Only very few studies started recently to also provide target-based runlengths when comparing multiobjective optimization algorithms and it is the main purpose of this paper to push forward this idea for benchmarking especially evolutionary multiobjective optimization algorithms. Custodio et al. [3] for example show performance profiles for the so-called purity indicator and two spread metrics as well as data profiles for purity while stochastic algorithms are displayed with respect to their worst, average, or best runs instead of displaying the full data distribution. In this benchmarking study, 100 test problems with different numbers of variables (from 2 till 30) and different numbers of objectives (2 and 3) are mixed within a plot while problems with lower search space dimension and less objectives are overrepresented. Denysiuk et al. [5] mainly follow the standard way of showing ε- and hypervolume indicator values after a given budget of function evaluations and use performance profiles only briefly to show the distributions of the median values of both indicators. In another study, Denysiuk et al. [6] also use performance profiles to study the median values of the IGD indicator. Also in those two papers, the performance profiles aggregate over problems with different search space and objective space dimensions.

When looking at these few studies that use data and performance profiles for comparing multiobjective optimization algorithms, four concerns can be raised:

- The first study uses quality indicators that are not compliant with the Pareto dominance relation. This is also the case for the CEC'2009 competition [20].
- All studies mix problems of different search space dimensions and numbers of objective functions which does not allow for scalability analyses.
- On the contrary, typically different problem accuracies (or target difficulties) are not combined in a single plot, although this seems to make sense as on a single function, this allows to rediscover the algorithms' convergence plots (see also the next section for details).
- Most of the used test problems have characteristics for which it is at least questionable whether they are representative for real-world problems (e.g. are many of the often used problems separable or they contain distinctive distance- and position-related variables).

While benchmarking studies coming from the EMO field, such as the CEC'2007 competition [12], do typically use Pareto-dominance-compliant quality indicators, most of the recent studies in EMO report the achieved quality indicator values for a few (arbitrary) numbers of function evaluations instead of reporting (several) target-based runlengths. Many studies furthermore report results for a disproportionately high number of separable functions. Also scalability analyses with respect to the number of variables are rare though scalability with respect to the number of objectives gained recent interest, see for example [17]. As to the benchmark problems, several well-known test suites such as the DTLZ, WFG, and CEC 2009 problems (see e.g. [9]) are available and accepted in the community, but besides their mentioned deficiencies (separability, distance- vs. position-related variables, ...), they do not naturally suggest the distinction between problems and instances and, hence, their usage might have resulted in overfitting algorithms' performances to the few available problems in each benchmark suite.

In the case of single-objective blackbox optimization, most of the above concerns have been addressed by the recent and nowadays well-established Blackbox Optimization Benchmarking (BBOB) framework [10]. Here, aggregation of data is never performed over different search space dimensions in order to allow for statements on the scalability of algorithms with increasing dimension whereas runtimes for more than one target value are typically aggregated in single plots. Moreover, a large effort has been spent on the choice of the test problems in order to select 24 meaningful and yet understandable test problems which all offer a large number of randomly generated instances[3]. The problems exhibit a wide range of difficulties observed in practice. Except for one, no BBOB problem is fully quadratic and only five of them are separable.

Our Contributions.

Here, we build upon this BBOB framework by (i) using combinations of the 24 single-objective BBOB test functions to create bi-objective test problems, by (ii) recording runlengths to reach predefined target difficulties in terms of quality indicator differences to a reference set, and finally by (iii) using data profiles to illustrate their distribution while aggregating over target values, problems, and indicators. Further details on the experimental setup will be given in the following sections.

In order to showcase the idea of recording target-based runlengths and providing data profiles to visualize them, we compare three common EMO algorithms on a set of 300 BBOB-based bi-objective test problems and illustrate which new insights can be gained by the new benchmarking approach in Sec. 5.2. We thereby advocate the usage of an *external archive* of all non-dominated solutions found by an algorithm so far, as we believe this is relevant practice in bi-objective real-world applications where function evaluations are expensive. The usage of the external archive to define an algorithm's quality has the additional advantage that algorithms of different population sizes can be easily compared with each other.

Let us note that, by no means, our proposal is meant as a sole replacement of any current benchmarking exercise, but is expected to be complemented by other benchmarking and visualization efforts such as empirical attainment functions [8] or a rigorous (automated) identification of the algorithms' "search controls and failure modes" [9].

2. DATA PROFILES

When benchmarking (single-objective) blackbox optimization algorithms, the first step is typically to define a test problem which consists of (i) a test function such as Sphere, Rosenbrock, etc., (ii) a specific instance of this test function, e.g., a shifted or rotated version of the original test function, and (iii) a given target difficulty f_{target} to which the instance of the test function is supposed to be solved. Then, we run our algorithms of interest on this test problem and report and compare the algorithms' runtimes in terms

[3]In addition to these 24 noiseless problems, a benchmark suite with 30 noisy BBOB functions exists.

of the number of function evaluations until the problem's target function value is reached.[4]

Most of the time, we do not have a single test problem we are interested in but a whole test suite and if the number of test problems is large, it might be tedious to look at the single runtimes for each problem. Then, it often makes sense to aggregate the results in one way or the other. One standard way of doing this graphically is to display so-called *data profiles* [16]. A data profile is an empirical cumulative distribution function (ECDF) of the observed runtimes over all test problems in which we can read (on the y-axis) how many problems have been solved by each algorithm for a given budget (on the x-axis)[5]. More formally, a data profile for a solver s on a problem class $\mathcal{P}$ is defined as

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|}\text{size}\left\{p \in \mathcal{P} \;\middle|\; \frac{t_{p,s}}{n_p} \leq \alpha\right\} \tag{1}$$

with $t_{p,s}$ being the observed runtime of solver s on problem p and n_p the number of variables of problem p [16]. Note that one can interpret the choice of p in Eq. 1 in two ways: either the problems are picked uniformly at random from $\mathcal{P}$ or the set of problems $\mathcal{P}$ is fixed beforehand and hence the choice of p is not always independent (for example if more than one target-based runlength is recorded for the same algorithm run). In the limit, however, both interpretations are equivalent. The idea behind having different instances of the same test function in a problem class is to allow for repetitions and statistically relevant results with respect to invariance properties of both stochastic and deterministic algorithms. Instead of running the same algorithm several times on the same instance, typically one (stochastic) algorithm run is performed on each instance, see e.g. [10].

The data profile approach has not only the advantage to display a large amount of data in a single plot that makes it easier to compare algorithms quickly (an algorithm is better if its data profile is more to the left and higher), but has also other important advantages:

Rediscovering convergence graphs: On a single function and when using a set of target values, plotting the data profile of a single run allows to rediscover the convergence graph (i.e. the best-so-far function value over time). The more target function values are defined, the closer the (vertically flipped) data profile is to the original convergence graph (where possibly the y-axis has to be rescaled accordingly). An illustration is shown in Fig. 1.

No constraint on the maximum number of function evaluations: Another beauty of the data profile approach is that algorithms can be compared even if they have not been run with the same number of function evaluations. The comparison is valid for runlengths up to the smallest budget among the compared algorithms.

Extracting information about function classes: Grouping different functions with similar properties to problem classes allows to extract information about on which types of problem algorithms are better and on which they are worse. In the single-objective BBOB testbed [10, 11], groups of separable or multimodal functions, for example, have been defined—between which one can see significant differences among the algorithms' performances.

All in all, data profiles are an important tool in state-of-the-art single-objective blackbox optimization benchmarking exercises that have been used already in a few derivative-free multiobjective optimization papers as indicated above. We are convinced that data profiles will also ease algorithm comparisons in the EMO field and consequently will lead to improved algorithms in practice. It will especially allow us to compare the performance of multiobjective algorithms over time while currently most studies (still) only look at a single or a few arbitrary budgets and report the quality values reached at those specific times. The additional advantage of displaying target-based runtimes is that the shown numbers are actually meaningful: It is easy to grasp that an algorithm that needs half as many function evaluations to solve a given problem is twice as efficient while knowing that the reached quality after the same budget is by a certain factor larger for one algorithm is usually difficult to interpret.

3. TRANSFERRING SINGLE-OBJECTIVE BENCHMARKING CONCEPTS TO THE MULTIOBJECTIVE SCENARIO

All we have to do to transfer the idea of data profiles to the multiobjective case is to define $t_{p,s}$ in Eq. 1 and, thus, more precisely what we define as a problem to be solved.

In the multiobjective case, k objective functions $f_i : X \to \mathbb{R}$ have to be simultaneously optimized (we assume, w.l.o.g., minimization here). If we assume the standard Pareto dominance relation $\prec$ is the underlying preference structure[6], the goal is to find (a subset or an approximation of) the Pareto set $\mathcal{P}_S = \{x^* \mid \nexists x \in X : x \prec x^*\}$. As usual, we call the image of $\mathcal{P}_S$ under $f = (f_1, \ldots, f_k)$ the Pareto front. Finding a subset or an approximation of the Pareto set is underspecified as such. Hence, we are typically interested in finding a solution set of as high quality as possible in as few function evaluations as possible where the quality of a solution set is given by a so-called quality indicator $I : 2^X \to \mathbb{R}$.

As a related issue, let us briefly discuss the difference between bounded and unbounded solution sets here. If we are interested in solution sets of a fixed size (of let us say μ), the optimization goal is to find a set of μ solutions among all sets of μ solutions that maximizes the quality indicator. For these optimal sets of μ solutions with respect to a quality indicator, the term *optimal μ-distribution* has been introduced [1]. If, on the other hand, an unbounded set with maximal quality indicator is sought, a good idea is to maintain an external archive of all non-dominated solutions found so far which, recently, even has been argued for when the the optimal μ-distribution is sought and can be extracted from the archive efficiently [2, 13].

Independent of whether an external archive is used or not, a problem p to define $t_{p,s}$ in the multiobjective case now consists of a given function per objective, a specific function instance per objective, a given quality indicator, and the target quality of the quality indicator. The quality indicator can thereby be computed with respect to a bounded or unbounded archive or as the indicator value of the best solution subset of size μ found so far. The value $t_{p,s}$ is then defined as the time to reach the given target indicator quality of problem p for solver s and the data profile can be plotted as in the single-objective case.

[4]We report ∞ if the target has not been reached in the experiment.

[5]Solving a problem thereby means that the algorithm run has reached the problem's target precision.

[6]We say solution x dominates solution y if $\forall 1 \leq i \leq k : f_i(x) \leq f_i(y)$ and $\exists 1 \leq i \leq k : f_i(x) < f_i(y)$ and write $x \prec y$.

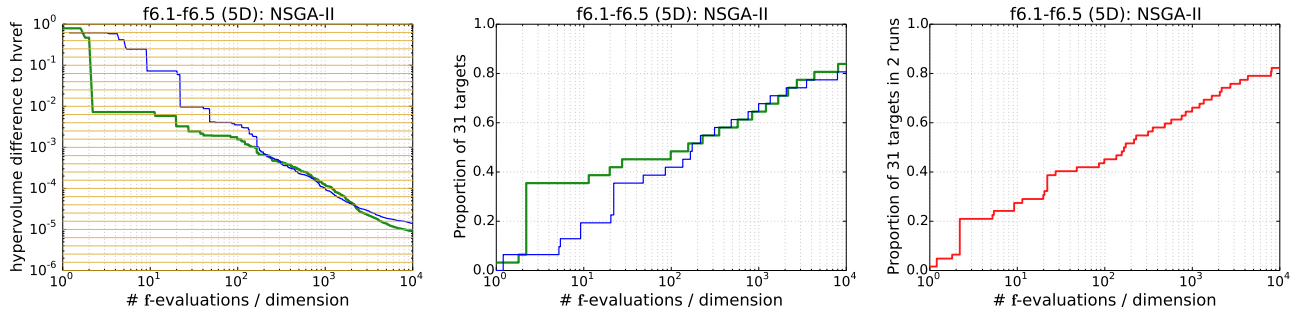

Figure 1: Recovery of convergence graphs using data profiles. The left plot shows the difference over time between the hypervolume indicator in two runs of NSGA-II to the hypervolume of the reference set on a combination of two instances (1 and 5) of the Attractive Sector function (f_6) in 5-D. The middle plot shows the corresponding data profile for each run individually using the 31 target values displayed as light (brown) horizontal lines on the left plot. The right plot shows the data profile, aggregated over the two runs.

As such, *multiobjective* data profiles do not only allow to aggregate results over several test functions and target difficulties but also over different quality indicators. However, one has to be careful that aggregating over different difficulties of the problems might bias the conclusions drawn from a data profile. This is evident when combining problems with different numbers of objective functions or search space dimensions—in particular if they are not uniformly distributed over the chosen problems. But also when combining results for several quality indicators, one should be careful to not introduce a strong bias, e.g., one should prevent that for one indicator significantly more target values are reached than for another indicator.

To not bias the implications made from a benchmarking exercise and to keep the test problems as close to real-world applications as possible is always an important and challenging task when designing a test problem suite. Hence, we rely here on an already established single-objective benchmark suite, the BBOB test problems [10, 11] and combine them to bi-objective problems by simply using all possible function combinations. Moreover, we use three well-known multiobjective optimization algorithms, namely NSGA-II [4], MOEA/D [19], and MO-CMA-ES [18], as a baseline to produce a reference set per problem instance, according to which the quality indicators are compared and the target difficulties are defined relatively. In the following, we will give some more details on this new multiobjective testbed.

4. GENERALIZING THE BBOB TESTBED TO MULTIOBJECTIVE OPTIMIZATION

The noiseless BBOB testbed [10] has 24 single-objective functions that are categorized into five groups: (1) five *separable* functions, (2) four *low or moderate condition* functions, (3) five *high condition* functions, (4) five *multi-modal* functions, and (5) five *multi-modal with weak global structure* functions. All functions are scalable in the number of decision variables. Typically, the testbed is used in dimensions 2, 3, 5, 10, 20, and sometimes 40. Importantly, all optima are known. Each of the functions has different *instances* in each dimension which are characterized by specific (random) transformations of the original function such as non-linear variable transformations or a rotation of the search space.

In order to generalize the noiseless BBOB testbed to a bi-objective one, and assuming that typical algorithms are invariant under permutations of the objective functions, we use all combinations of two BBOB functions f_i and f_j such that $1 \leq i \leq j \leq 24$ which gives $\binom{24}{2} + 24 = 300$ function combinations. The bi-objective instances are generated by choosing non-repetitive instances of the single-objective functions with the additional constraint that the two corresponding single-objective optima are different for each bi-objective function combination (across all 300 combinations and over dimensions 2, 3, 5, 10, 20, and 40). Here, we propose combining the five instances 2, 3, 7, 9, and 11 of the first objective function pairwisely with the five instances 4, 5, 8, 10, and 12 of the second objective function which guarantees that the single-objective optima are always different.

The above mentioned five function groups of the single-objective BBOB testbed result in natural—with the same argumentation as above—$\binom{5}{2}+5 = 15$ groups of bi-objective test functions of which the first objective is chosen from a first function group and the second objective function is chosen from a second (potentially the same) function group. Since all but one function group have five elements, the 15 bi-objective function groups contain either $5 \cdot 5 = 25$ (10 groups), $5 \cdot 4 = 20$ (4 groups), or $4 \cdot 4 = 16$ (1 group) bi-objective problems. We have now, for example, a group of *separable–separable* bi-objective problems where separable functions are combined together or a *separable–multi-modal* group. Additionally, we can aggregate over all possible combinations of single-objective test functions, which will be denoted as *all functions–all functions* in the following plots.

Now we have a suite of multiobjective test problems with instances; what is missing is to define quality indicators and target values. As to the former, we use the two most popular and often recommended quality indicators of hypervolume and (additive) ε-indicator [22]. More precisely, we use the difference between the unary hypervolume indicator of the set of all non-dominated solutions found so far (in other words, the *archive*) and the hypervolume of a pre-computed *reference set* as well as the binary ε-indicator between the archive and the reference set. All objective vectors are scaled in the performance assessment step such that the known ideal and nadir points are at $(0, 0)$ and $(1, 1)$ respectively. At every new function evaluation, the archive and the hypervolume indicator difference are updated efficiently *on-the-fly*. The ε-indicator, for the time being, is computed from the stored non-dominated archive in a postprocessing step though a similar on-the-fly calculation is possible.

To compute the reference set for the indicators, we ran the algorithms NSGA-II (15 independent trials), MOEA/D (15 trials), and MO-CMA-ES (10 trials)[7] with population sizes of 1000, 1000, and 200 for $5{\cdot}10^5 D$, $5{\cdot}10^5 D$, and $10^5 D$ function evaluations, respectively. The non-dominated set of all evaluated solutions per test function instance is then used as the reference set.

As to the target values, 70 linearly spaced targets in the logarithmic scale from $10^{-0.1}$ to 10^{-7} were used for the hypervolume indicator and from $10^{-0.1}$ to 10^{-5} for the ε-indicator. To save disk space, we have used a thinning strategy that only writes *new* non-dominated solutions accumulated in the memory to disk when a new hypervolume target is hit. Non-dominated solutions generated after the previous target hit that are dominated at the time of the current target hit are explicitly discarded.

The usage of the external archive to define the performance of an algorithm in the bi-objective case is probably the most arguable part of our proposed benchmarking. However, note that in the bi-objective case, all obtained non-dominated objective vectors can be plotted easily to get an idea of the Pareto front and second, this approach allows to compare algorithms with different population sizes naturally. Moreover, the implementation of the archiving as well as the on-the-fly computation of the two indicator values is relatively cheap in the bi-objective case and does not add up much within the benchmarking (see Sec. 5.2). Note finally, that in the future, we plan to include also the extracted best subset of the archive of a specific size as performance measure, similar to [2, 13].

The proposed multiobjective BBOB test suite was implemented upon the COCO (COmparing Continuous Optimisers, [10]) platform and is available for download at http://coco.gforge.inria.fr/GECCO2015-MOBBOB/.

5. CASE STUDY ON THE MULTIOBJECTIVE BBOB TESTBED

In the remainder of the paper, we showcase what can be extracted from recorded data and demonstrate how the proposed multiobjective BBOB, by using data profiles, distinguishes from previous multiobjective benchmarking studies.

5.1 Experimental Setup

The three algorithms NSGA-II, MOEA/D, and MO-CMA-ES have been benchmarked on the proposed multiobjective testbed. Each algorithm was run once on each of the above mentioned 5 instances on every of the 300 bi-objective problems in dimensions 2, 3, 5, 10, and 20. All algorithms used a population size of 200. Since the employed implementations of NSGA-II and MOEA/D turned out to be fast compared to the Shark implementation of MO-CMA-ES, we allocated the same $10^5 D$ function evaluations for NSGA-II and MOEA/D while limiting the latter to $4{\cdot}10^4 D$ evaluations.

It should be noted that NSGA-II used the simulated binary crossover with a distribution index of 20 and a crossover rate of 1 and polynomial mutation, having a distribution index of 50 with a mutation rate of $1/D$. MOEA/D employed the differential evolution crossover of [14] with parameters $CR = 0.8$ and $F = 0.9$, and exactly the same mutation as of NSGA-II. In MOEA/D, a neighborhood size of 20 is used with a probability of 0.8 for selecting mating parents from the neighborhood. MO-CMA-ES uses the population-based notion of success within the generational evolutionary strategy (the $(\mu+\mu)$MO-CMA-ES^P version as in [18]).

We have also conducted an additional experiment to demonstrate the impact of the population size on the performance of NSGA-II, to which end we ran NSGA-II with population sizes of 48, 100, 148, 200, 500, and 1000 on the 5-D problems. All configurations were run up to $4{\cdot}10^5 D$ function evaluations except for those with large populations of 500 and 1000 individuals, which were run for $2{\cdot}10^5 D$ and $10^5 D$ function evaluations, respectively.

5.2 Experimental Results

The following paragraphs describe the results obtained from the benchmarking as displayed in Figures 2–6 and discuss the advantages gained over previous benchmarking efforts without the use of the archive, target-based runlengths, and data profiles. Due to space limitations, not all plots can be shown here but additional material is provided at http://coco.gforge.inria.fr/GECCO2015-MOBBOB/.

Possibility to Compare Algorithms with Different Population Sizes and Different Stopping Criteria: Figure 2 provides examples of data profiles displaying the performance of NSGA-II with different population sizes on three 5-D test functions. We clearly see that NSGA-II, when using a larger population size, becomes consistently slower in the beginning of the optimization (roughly by a factor of 6 between population sizes 48 and 1000). This is mostly related to the fact that the entire first generation resembles random search. The performance in the later stages of the search are less clear: whereas on the f_{12}–f_{21} combination, a larger population size clearly helps, the f_5–f_{11} example shows no trend. Note also that the algorithms displayed in Fig. 2 have been run for different numbers of function evaluations (indicated by big crosses at the evaluation budgets) which is a clear advantage of using the indicator values of all non-dominated solutions found as performance criterion.

Influence of Search Space Dimension: Figure 3 presents the data profiles of NSGA-II, MOEA/D, and MO-CMA-ES over five instances on the combinations of separable Ellipsoid (f_2) and separable Rastrigin (f_3) functions in dimensions 2, 3, 5, 10, and 20. Due to lack of space, only the individual-dimension plots for 2-D, 5-D, and 20-D are shown. It is, however, sufficient to realize the different behavior of the algorithms across dimensions. Accordingly, the aggregation over dimensions, as shown in the rightmost plot, turns out to be less interpretive. Indeed, without looking at the 2-D plot, it is hard to know from the aggregation plot that the hypervolume of MO-CMA-ES begins to approach that of the other algorithms in the 2–$4{\cdot}10^4$ functions evaluations range in 2-D. Because performance differences among algorithms might change drastically over dimensions, showing data profiles that aggregate over dimensions is usually not recommended. More informative, on the contrary, are scaling plots such as the ones in Fig. 4. Here, the expected running times [10] to hit the hypervolume target of 10^{-3} are plotted for all algorithms across five dimension values. NSGA-II is seen to scale better than the other algorithms on

[7] We employed the original implementations of NSGA-II in C from http://www.iitk.ac.in/kangal/codes.shtml, MOEA/D in C++ from http://dces.essex.ac.uk/staff/zhang/webofmoead.htm, and MO-CMA-ES in C++ from the Shark library at http://image.diku.dk/shark.

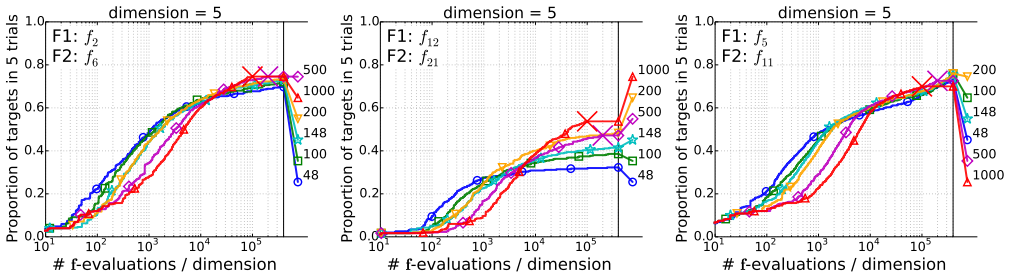

Figure 2: Comparison of six NSGA-II configurations using data profiles based on the hypervolume indicator. Each configuration goes with a different population size in $\{48, 100, 148, 200, 500, 1000\}$ and some of them have different run lengths. Cross signs indicate the function evaluation at which the algorithm was terminated.

the separable f_2-f_3 combination and solves the 20-D problem up to the given target while the other algorithms never hit this target within their allocated budget. Note that this behavior is specific to the separable functions shown here due to the fact that NSGA-II and MOEA/D's variation operators exploit separability while the MO-CMA-ES is invariant under rotations of the search space. Indeed, when a separable function is combined with a rotated, a high-condition, or a multi-modal function (Fig. 4, from left to right), the MO-CMA-ES is seen to scale significantly better than NSGA-II and MOEA/D, especially in the last case where only the MO-CMA-ES can solve the 10-D problem.

Aggregating Data over Test Functions: Looking at every single function combination and dimension is tedious, especially when 300 function combinations are involved. Thus, it is advisable to aggregate the data over all functions or a subset thereof to get a quick idea about performance differences first. Data profiles make this easy as can be seen in Fig. 5. The aggregation over all 300 function combinations looses most information (left) while the aggregation over functions with the same properties (middle and right) seems to be a good compromise and results in more pronounced, yet interpretable results. The overall picture in the left plot of Fig. 5, for example, tells us that NSGA-II outperforms the other two algorithms until about $6 \cdot 10^3$ function evaluations at which MO-CMA-ES takes the lead. MOEA/D seems dominated by NSGA-II at any budget of function evaluations. We can see that this is not the full picture if we focus on aggregations over function groups. If all objective functions are separable (Fig. 5, middle), the conclusions change drastically: NSGA-II becomes the dominating algorithm, outperforming the other two at any time. On the combination of moderate and multi-modal functions, however, even MOEA/D starts to outperform NSGA-II from around $5 \cdot 10^3$ function evaluations.

Aggregating Over Different Quality Indicators: Finally, data profiles also allow to aggregate performance over different indicators. In Figure 6, we show data profiles aggregating over hypervolume and ε-indicator on 5-D problems. The left-most and middle plots show the data profiles for individual indicators while the right-most plot shows the aggregated data over both indicators—exemplary for the combination of the Rosenbrock (f_8) and Weierstrass (f_{16}) functions. We observe that MO-CMA-ES is faster than the other two algorithms after about $4 \cdot 10^4 D$ function evaluations w.r.t. either of the two quality indicators. The superiority of MO-CMA-ES is then confirmed in the data profiles aggregating over both indicators.

Few Details on Timing, Data Storage and Thinning: Maintaining the archive of nondominated solutions and the computation of the indicators naïvely can be time consuming. In the case of only two objectives, however, both the archive and the hypervolume indicator value can be updated on-the-fly using appropriate data structures. Our implementation (in C) thus allows us to keep the additional overhead of the benchmarking small. Under a single core of the Intel i7-3520M CPU @ 2.9 GHz, running NSGA-II on an instance of the 300 combinations over 10^4 function evaluations, for example, takes 110 seconds without data logging, 119 seconds with logging nondominated solutions over time, and 120 seconds with logging nondominated solutions and computing the hypervolume indicator on-the-fly.

Once the actual algorithm runs are complete and the raw data is logged, a postprocessing script in Python enables us to automatically produce the plots as presented in this paper for the visual comparison of the algorithms. This script basically performs two major functions: processing the logged data to create a database of data profiles and plotting (and aggregating) these data profiles. Most of the postprocessing time lies in the extraction of data profiles. It takes, for example, about 12 minutes to build all data profiles for NSGA-II in 5-D and 3 minutes for the plotting.

Although the proposed data thinning results in significantly smaller data files than if all non-dominated points are recorded (typically 30% less), the amount of produced data for a typical benchmarking experiment is still quite large. This is unfortunately unavoidable if the unbounded external archive has to be stored. Running MO-CMA-ES resulted in the largest amount of data among the three tested algorithms. The corresponding amount of logged data when running MO-CMA-ES on five instances of all 300 test problems in dimensions 2, 3, 5, 10, and 20 is 3.8 GB compared to 2.2 GB for NSGA-II and 2.4 GB for MOEA/D. Though this amount is reasonable, given the cheap storage capacities these days, we plan to reduce this size further to ease the online distribution of algorithm data sets and to improve the ability to benchmark larger sets of algorithms.

6. CONCLUSIONS

Reporting the empirical cumulative distribution of target-based run lengths in so-called data profiles is a standard way to display and compare the performance of single-objective optimizers. Contrary, in the multiobjective case, most of the time, some statistics about the reached quality indicators for a certain budget (or a few budgets in number of function evaluations) are reported. In this paper, we argue in favor

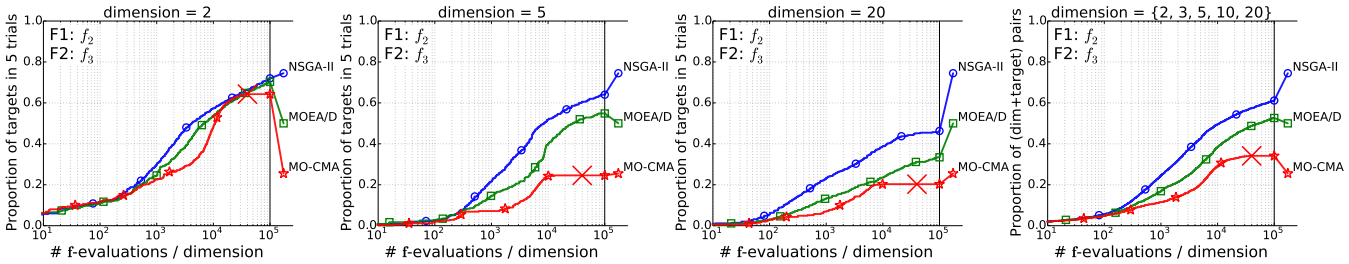

Figure 3: Influence of search space dimensions on the combination of separable Ellipsoid (f_2) and separable Rastrigin (f_3) functions w.r.t. the hypervolume indicator. The rightmost plot shows the aggregation of data profiles over all dimensions. Cross signs indicate the budget for MO-CMA-ES.

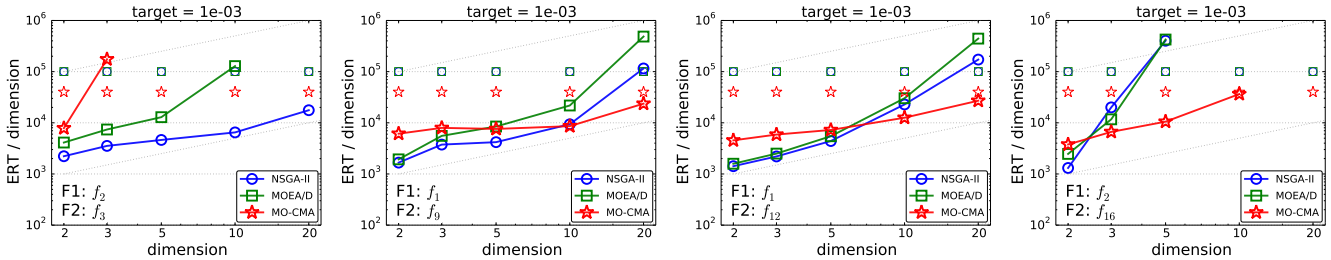

Figure 4: Expected running time (ERT, in number of f-evaluations) divided by dimension for the hypervolume target value 10^{-3} versus dimension. Light symbols give the maximum number of f-evaluations from the longest trial divided by dimension. Horizontal lines give linear scaling and slanted dotted lines give quadratic scaling.

of using data profiles also for benchmarking multiobjective optimizers and show how this can be achieved efficiently when combining the single-objective BBOB functions of [10, 11] to a bi-objective test suite. The proposed approach to performance assessment has several advantages:

- Reporting target-based runlengths instead of budget based quality indicator values makes the displayed values meaningful: it is for example easier to grasp that an algorithm takes twice as long to solve a certain problem than to understand what a quality indicator difference of a certain value means.
- Measuring the quality of an algorithm via an external archive allows to compare methods with different population sizes.
- Displaying empirical cumulative distribution functions aka data profiles allows to evaluate the algorithms over time (in an anytime setting) while no maximum number of function evaluations has to be prescribed.
- Using data profiles enables the aggregation of results over different targets, over different functions or function groups, and over different quality indicators.
- The use of well-established and well-understood single-objective test functions can help to explain performance differences of multiobjective optimizers but also to improve currently available algorithms for tackling the difficulties observed in practical applications.

The main disadvantage of the approach and the biggest challenge is that meaningful target f-values need to be defined to get meaningful problems.

Exploiting the described methodology for benchmarking state-of-the-art algorithms extensively and for designing new algorithms will be one of the obvious next steps. Extensions towards other quality measures that can be efficiently computed on-the-fly are planned. It is certainly interesting to additionally use the highest indicator value over all subsets of the archive with exactly μ solutions as a performance criterion as suggested in [2, 13].

Acknowledgments This work was supported by the grant ANR-12-MONU-0009 (NumBBO) of the French National Research Agency. TDT is further supported by an individual PhD grant of Inria.

7. REFERENCES

[1] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Theory of the Hypervolume Indicator: Optimal μ-Distributions and the Choice of the Reference Point. In *Foundations of Genetic Algorithms (FOGA 2009)*, pages 87–102. ACM, 2009.

[2] K. Bringmann, T. Friedrich, and P. Klitzke. Two-dimensional Subset Selection for Hypervolume and Epsilon-Indicator. In *Genetic and Evolutionary Computation Conference (GECCO 2014)*, pages 589–596. ACM Press, 2014.

[3] A. L. Custódio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente. Direct multisearch for multiobjective optimization. *SIAM Journal on Optimization*, 21:1109–1140, 2011.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[5] R. Denysiuk, L. Costa, and I. E. Santo. A New Hybrid Evolutionary Multiobjective Algorithm Guided by Descent Directions. *Journal of Mathematical Modelling and Algorithms*, 12:233–251, 2013.

[6] R. Denysiuk, L. Costa, and I. E. Santo. Many-Objective Optimization using Differential Evolution with Variable-Wise Mutation Restriction. In *Conference on Genetic and Evolutionary Computation (GECCO 2013)*, pages 591–598. ACM, 2013.

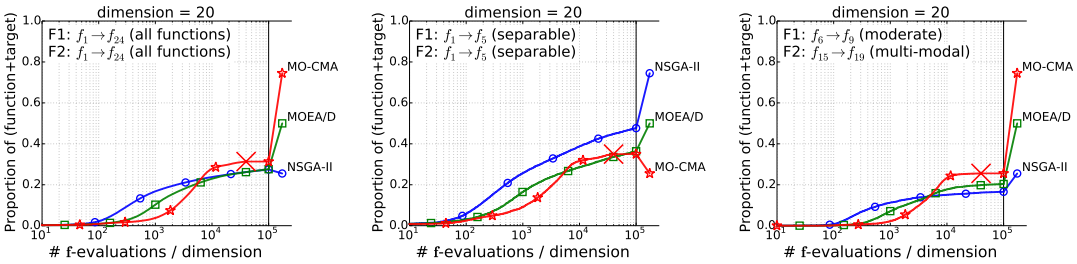

Figure 5: Hypervolume-indicator-based data profiles aggregating over all 300 problems (left), over all combinations of separable functions (middle), and over all combinations of moderate and multi-modal functions.

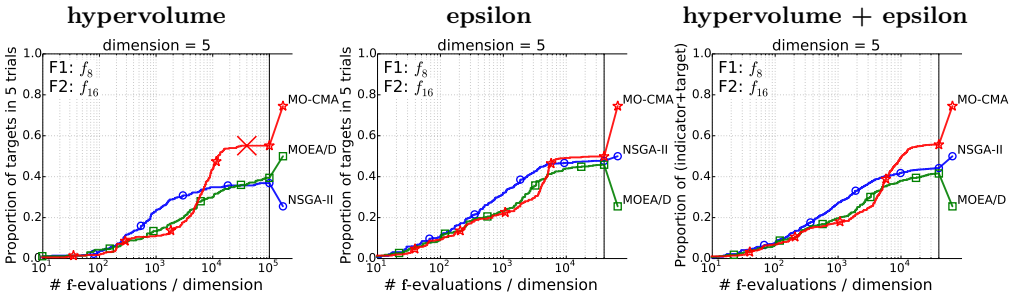

Figure 6: Data profiles of algorithm performances for the combination of Rosenbrock (f_8) and Weierstrass (f_{16}) functions in 5-D. The left and center plots show data profiles for the hypervolume and ε-indicator respectively, and the right plot shows the aggregation over both indicators.

[7] E. D. Dolan and J. J. Moré. Benchmarking Optimization software with Performance Profiles. *Mathematical Programming*, 91:201–213, 2002.

[8] V. Grunert da Fonseca, C. M. Fonseca, and A. O. Hall. Inferential Performance Assessment of Stochastic Optimisers and the Attainment Function. In *Evolutionary Multi-Criterion Optimization (EMO 2001)*, pages 213–225. Springer, 2001.

[9] D. Hadka and P. Reed. Diagnostic Assessment of Search Controls and Failure Modes in Many-objective Evolutionary Optimization. *Evolutionary Computation*, 20(3):423–452, 2012.

[10] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-Parameter Black-Box Optimization Benchmarking 2009: Experimental Setup. Research Report RR-6828, INRIA Saclay, 2009.

[11] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Posík. Comparing Results of 31 Algorithms from the Black-box Optimization Benchmarking BBOB-2009. In *GECCO workshop on Black-Box Optimization Benchmarking (BBOB'2010)*, pages 1689–1696, 2010.

[12] V. L. Huang, A. K. Qin, K. Deb, E. Zitzler, P. N. Suganthan, J. J. Liang, M. Preuss, and S. Huband. Problem Definitions for Performance Assessment of Multi-objective Optimization Algorithms. Technical report, Nanyang Technological University, 2007.

[13] T. Kuhn, C. M. Fonseca, L. Paquete, S. Ruzika, and J. R. Figueira. Hypervolume Subset Selection in Two Dimensions: Formulations and Algorithms. Technical report, University of Kaiserslautern, 2014.

[14] S. Kukkonen and J. Lampinen. Performance Assessment of Generalized Differential Evolution 3 with a Given Set of Constrained Multi-objective Test Problems. In *IEEE Congress on Evolutionary Computation (CEC 2009)*, pages 1943–1950, 2009.

[15] M. López-Ibáñez, L. Paquete, and T. Stützle. Exploratory Analysis of Stochastic Local Search Algorithms in Biobjective Optimization. In *Experimental Methods for the Analysis of Optimization Algorithms*, pages 209–222. Springer, 2010.

[16] J. Moré and S. Wild. Benchmarking Derivative-Free Optimization Algorithms. *SIAM J. Optimization*, 20(1):172–191, 2009.

[17] C. von Lücken, B. Barán, and C. Brizuela. A Survey on Multi-Objective Evolutionary Algorithms for Many-Objective Problems. *Computational Optimization and Applications*, 58(3):707–756, 2014.

[18] T. Voß, N. Hansen, and C. Igel. Improved Step Size Adaptation for the MO-CMA-ES. In *Conference on Genetic and Evolutionary Computation (GECCO 2010)*, pages 487–494. ACM, 2010.

[19] Q. Zhang and H. Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[20] Q. Zhang, A. Zhou, S. Zhao, P. Suganthan, W. Liu, and S. Tiwari. Multiobjective Optimization Test Instances for the CEC 2009 Special Session and Competition. Technical report, Univ. of Essex, 2009.

[21] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH Zurich, Switzerland, 1999.

[22] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.